PROCESSING TECHNIQUES FOR SERVERS HANDLING CLIENT/SERVER TRAFFIC AND COMMUNICATIONS

ABSTRACT OF THE DISCLOSURE

The present invention relates to a system for handling client/server traffic and communications pertaining to the delivery of hypertext information to a client. The system includes a central server which processes a request for a web page from a client. The central server is in communication with a number of processing/storage entities, such as an annotation means, a cache, and a number of servers which provide identification information. The system operates by receiving a request for a web page from a client. The cache is then examined to determine whether information for the requested web page is available. If such information is available, it is forwarded promptly to the client for display. Otherwise, the central server retrieves the relevant information for the requested web page from the pertinent server. The relevant information is then processed by the annotation means to generate additional relevant computer information that can be incorporated to create an annotated version of the requested web page which includes additional displayable hypertext information. The central server then relays the additional relevant computer information to the client so as to allow the annotated version of the requested web page to be displayed. In addition, the central server can update the cache with information from the annotated version. The central server can also interact with different servers to collect and maintain statistical usage information. In handling its communications with various processing/storage entities, the operating system running behind the central server utilizes a pool of persistent threads and an independent task queue to improve the efficiency of the central server. A task needs to have a thread assigned to it before the task can be executed. The pool of threads are continually maintained and monitored by the operating system. Whenever a thread is available, the operating system identifies the next executable task in the task queue and assigns the available thread to such task so as to allow it to be executed. Upon conclusion of the task execution, the assigned thread is released back into the thread pool. An additional I/O queue for specifically handling input/output tasks can also be used to further improve the efficiency of the central server.